山东大学
SHANDONG UNIVERSITY

崇新学堂

2024 — 2025 学年第一学期

实验报告

课程名称：Introduction to EECS Lab

实验名称：DL10 - Robot Pets

学生姓名：胡君安、陈焕斌、黄颢

实验时间：2024 年 12 月 5日

DL10 - Robot Pets

1. Introduction

Design Lab 10 focuses on building and testing a controller a simple “pet robot.” We will:

• Demonstrate an integrated analog and digital controller for the robot to

follow a light.

• Integrate the sonar sensors with your controller.

2. Experimental step

Step 1:

We designed and implemented a robot behavior that uses signals from its head to turn the robot toward a bright light. By reading voltages from pins 1, 3, 5, and 7 of the robot connector through the `inp.analogInputs` list, we controlled the robot's rotational velocity (`rvel`). After debugging by tilting the robot backward and manually adjusting the head, we found that a gain of `0.1` in the control loop provided stable responses.

```
Python
class MySMClass(sm.SM):
    startState = 'stop'
    def getNextValues(self, state, inp):
        V01 = inp.analogInputs[1]   #pot
        V02 = inp.analogInputs[2]   #motor
        V_pot = 5.1
        gain_rotate = 0.5
        rotate_velocity = gain_rotate * (V_pot - V01)

        print V01
        print V02

        if V01 <= 4.8 or V01 >= 5.3:
            return ('rotate', io.Action(fvel=0, rvel=rotate_velocity))
        else:
            return ('stop', io.Action(fvel=0, rvel=0))
```

Step 2:

We aim to redesign the brain and circuit so that the robot's behavior is influenced by its proximity to the light. When the light is off, the robot should remain stationary. When the light is on, the robot should move towards it and stop at a distance of approximately half a meter from the bulb.

```
Python
```

```
class MySMClass(sm.SM):
    startState="turn"
    def getNextValues(self, state, inp):
        current_location = inp.analogInputs[1]
        light_intensity = inp.analogInputs[2]
        base_position = 5
        ref_light_level = 7.5
        threshold_high = 8
        threshold_low = 7
        print light_intensity

        if state == 'turn':
            if current_location == base_position:
                return("light", io.Action(fvel=0, rvel=0))
            else:
                return('turn', io.Action(fvel=0,
                                         rvel=0.5 * (base_position -
current_location)))

        if state == 'light':
            if threshold_low <= light_intensity <= threshold_high:
                return('turn', io.Action(fvel=0, rvel=0))
            elif light_intensity < threshold_low:
                return('light', io.Action(fvel=-1 * (light_intensity -
ref_light_level), rvel=0))
            elif light_intensity > threshold_high:
                return('light', io.Action(fvel=1 * (light_intensity -
ref_light_level), rvel=0))
```

We use the output voltage in the middle of the photosensitive resistor to measure the light intensity and judge the distance, so we decided to achieve the goal by connecting an additional resistor in series to divide the voltage--we attach a fixed resistance to one photoresistor in series.(We assume that both photoresistors have the same rate of change with light)

Because of the extremely fast response of the head to the light , we only consider the case where the light is in the middle, which means that the two photoresistors have the same resistance value.The divided voltage formula is $U_o = \frac{R_l}{2R_l+R} = \frac{1}{2+\frac{R}{R_l}}$. From the formula, we're able to know that the output voltage between two photoresistors changes monotonically with distance.They have an inverse relationship.

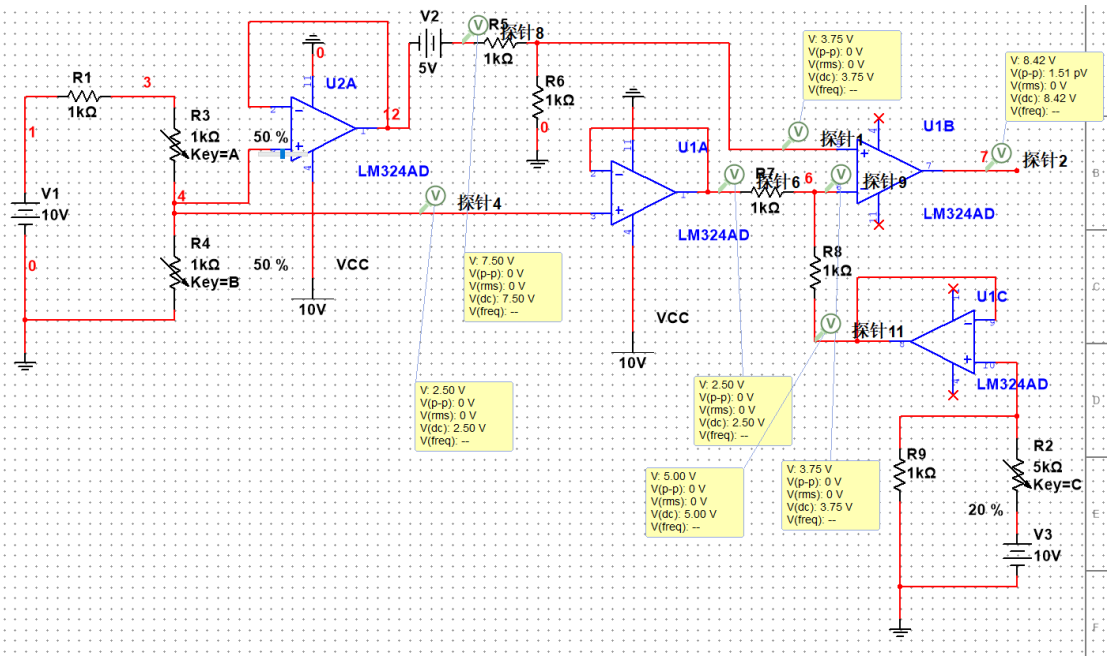
Then we mentioned a problem. With the change of the output voltage of the photoresistors, the voltage at the positive electrode of the motor is also changing correspondingly. We decided to solve it by adjusting the input reference voltage of the other end of the reverse amplifier, because the voltage of the positive electrode will vary with distance, the voltage in both schemes should also vary with distance.

The only thing we need to do is to ensure that $V\mathrm{o} = VM- = 5V$ when the resistance values of the two photoresistors are equal. From HW2, we can get the formula between

Vo and V

$$(V_0 - V) = -\frac{R_2}{R_1}(V_2 - V) = -(V_2 - V)$$

$$V = \left(\frac{R_l}{2R_l + R} + 5\right)/2$$



circuit diagram

Through this circuit diagram we achieved our goal.

Check Yourself 1

What control variable(s) from the head is/are important for determining proximity to the light?

The output voltage in the middle of the photosensitive resistor. The reason is described above.

Checkoff 1.

Wk.10.2.1: Demonstrate your pet robot's basic behaviors, including facing the light, approaching the light, retreating from it, and patiently waiting when there is no light. For extra brownie points, try parallel parking.

Due to a malfunction of the robot, this part cannot be verified.

Step 3:

We aim to enhance the robot's behavior by incorporating sonar sensors to achieve more dynamic interactions. Using state machine combinators, similar to those in Design Lab 3, we will construct a brain that integrates the boundary-following behavior from Design Lab 2 with the light-following controller. When the light is sufficiently bright and the path is clear, the robot will follow the light. Otherwise, it will adhere to the wall.

```
Python
from boundaryFollower import boundaryFollowerClass

class MySMClass(sm.SM):
    startState = 'stop'
    def getNextValues(self, state, inp):
        V_Location = inp.analogInputs[0]
        V_Light = inp.analogInputs[1]

        V_Base = 5
        V_0 = 7.5
        V_thh = 8
        V_thl = 7
        V_diffLocation = V_Base - V_Location
        V_diffLight = V_Light - V_0

        k_Location = 0.5
        k_Light = 1

        print(V_Light)

        if state == 'stop':
            if V_Location == V_Base:
                return ('light', io.Action(fvel=0, rvel=0))
            else:
                return ('turn', io.Action(fvel=0,
rvel=k_Location*V_diffLocation))

        if state == 'turn':
            if V_Location == V_Base:
                return ('stop', io.Action(fvel=0, rvel=0))
            else:
                return ('turn', io.Action(fvel=0,
rvel=k_Location*V_diffLocation))

        if state == 'light':
            if V_thl <= V_Light <= V_thh:
                return ('light', io.Action(fvel=0, rvel=0))
            elif V_Light < V_thl:
                return ('light', io.Action(fvel=k_Light*V_diffLight, rvel=0))
            elif V_Light > V_thh:
                return ('light', io.Action(fvel=k_Light*V_diffLight, rvel=0))

def Light(inp):
    V = inp.analogInputs[1]
    V_low = 10
    if V >= V_low:
        return False
    else :
        return True
def Distance(inp):
    sonars = inp.sonars
    if sonars[3] >=0.8:
```

```
        return True
    else :
        return False

mySM = sm.Switch(Light, sm.Switch(Distance, MySMClass(),
boundaryFollowerClass()),
                 boundaryFollowerClass())
mySM.name = 'brainSM'
```

3. Summary

- The robot's rotational velocity is managed by reading voltages from specific pins. After debugging, a 0.1 gain in the control loop stabilizes responses. Our code logic dictates rotation based on potentiometer voltage comparison with thresholds

- By measuring light intensity with a series-connected resistor to the photosensitive resistor, the robot's actions adapt to light presence. With our derived voltage formulas and adjustments to the reverse amplifier's reference voltage, our robot can approach or halt relative to the light source

- Incorporating sonar sensors, a composite state machine is crafted. Leveraging state machine combinators, it fuses boundary-following and light-following behaviors. Depending on light intensity and path clearance, the robot switches between light pursuit and wall adherence, demonstrating enhanced autonomous adaptability.